

Jazyk C

Paměť I

Mgr. Markéta Trnečková, Ph.D.



Palacký University, Olomouc

Zásobník

Halda

Globální
proměnné

Konstanty

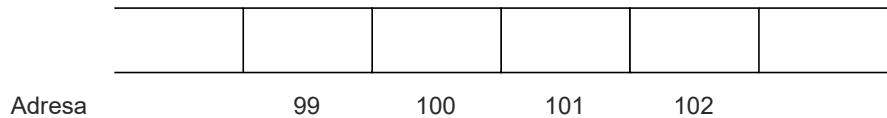
Kód

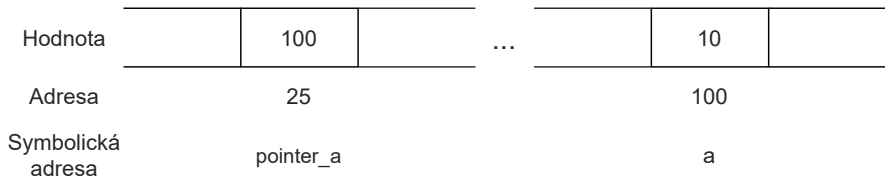


- deklarace proměnné
- definice proměnné
- rozsah platnosti identifikátoru



- **Statická alokace**
- **Dynamická alokace**
 - Na zásobník
 - Na haldu







deklarace:

```
typ *jmeno;
```

Nulový ukazatel: NULL

výpis:

```
printf("%p", ptr);
```

operátory:

- adresy: &
- dereference: *

Example

```
int a = 3, b, *ptr1, *ptr2;

/* ukazatel ptr1 ukazuje na promennou a */
ptr1 = &a;

/* hodnota b je 5 (*ptr1 je rovna 3) */
b = *ptr1 + 2;

/* ptr1 a ptr2 ukazují na stejné místo */
ptr2 = ptr1;

/* změníme hodnotu na místě, kam ukazuje ptr2 */
*ptr2 = 5;

/* hodnota b bude 8 */
b = a + 3;
```




- Statická pole
- Dynamicky alokovaná pole

Statické pole



deklarace:

```
typ jmeno[velikost];
```

definice:

```
typ jmeno[velikost] = {p1, p2, ..., pn};
```

```
typ jmeno[] = {p1, p2, ..., pn};
```

Example

```
/* Neinicializovane pole velikosti 6 */
```

```
int pole1[6];
```

```
/* Pole o velikosti 10, se 3 inicializovanymi prvky */
```

```
int pole2[10] = {1, 2, 3};
```

```
/* Pole o velikosti 4 se vsemi inicializovanymi prvky */
```

```
int pole3[] = {1, 2, 3, 4};
```

...	1. prvek	2. prvek	3. prvek	4. prvek	...
Index:	0	1	2	3	

jmeno [index] ;

Example (Která část kódu není správně?)

```
int pole [5] = {1, 2, 3, 4, 5};  
  
pole [2] = 10;  
pole [3] = pole [1] + pole [2];  
pole [1] = pole [5];  
pole [7] = 10;
```



Example

```
int pole[20];
int i;

for(i = 0; i < 20; i++){
    pole[i] = 2*i;
}
```

'a'	'h'	'o'	'j'	' '	's'	'v'	'e'	't'	'e'	'\0'
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------

definice:

```
char retezec[11] = {'a', 'h', 'o', 'j', ' ', 's', 'v', 'e', 't', 'e', '\0'};
```

```
char retezec[] = "ahoj svete";
```

Pozor!

```
char *retezec = "ahoj svete";
```

```
char retezec[5];  
retezec = "duha";
```

výpis:

```
printf("%s" , retezec );
```

vstup:

```
char text[11];  
scanf("%10s" , text );
```

Meze všech polí i řetězců budeš kontrolovati, neboť tam, kde ty použiješ slovo „test“, jiný zajisté napíše „naprostoneuveritelnedlouhoublost“.

(Henry Spencer)

Ostatní: string.h

- strcat()
- strstr()
- strcpy()



- součet pointeru a celého čísla
- rozdíl pointeru a celého čísla
- porovnání dvou pointerů (na stejný datový typ)
- rozdíl dvou pointerů (na stejný datový typ)



Example

```
char *ptr_c = 10; /* sizeof(char) = 1 */  
int *ptr_i = 10; /* sizeof(int) = 2 */  
float *ptr_f = 10; /* sizeof(float) = 4 */
```


Example

```
int pole[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int *ptr;
int i;

ptr = pole;

for (i = 0; i < 10; i++){
    /* hodnota prvku pole */
    printf("Hodnota *ptr = %d\n", *(ptr + i));
    /* adresa prvku pole */
    printf("Adresa ptr = %p\n\n", (ptr + i));
}
```

Example

```
int pole[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
int *ptr;
int i;

ptr = pole;

for (i = 0; i < 10; i++){
    /* hodnota prvku pole */
    printf("Hodnota *ptr = %d\n", *ptr);
    /* adresa prvku pole */
    printf("Adresa ptr = %p\n\n", ptr);
    ptr++;
}
```

Proč pole indexujeme od 0?



Example

```
int pole[] = {1, 2, 3};  
  
printf("1. prvek je %d \n", pole[0]);  
printf("1. prvek je %d \n", *pole);
```

Example

```
int pole[] = {1,2,3};  
  
printf("3. prvek je %d \n", pole[2]);  
printf("3. prvek je %d \n", *(pole+2));
```

Proč pole indexujeme od 0?



Example

```
int pole [] = {1, 2, 3};  
printf("%d", 2[pole]);
```

`pole[2] == (pole + 2) == (2 + pole) == 2[pole]`

Example

```
int pole [] = {1,2,3,4,5,6,7,8,9,10};
int *ptr;

for (ptr = pole; ptr < pole + 10; ptr++){
    /* hodnota prvku pole */
    printf("Hodnota *ptr = %d\n", *ptr);
    /* adresa prvku pole */
    printf("Adresa ptr = %p\n\n", ptr);
}
```



```
int pole[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
```

- 1 Jak zjistíte adresu 4. prvku pole?
- 2 Máme ukazatel, který ukazuje na některý prvek pole pole. Jak zjistíte index tohoto prvku v poli?
- 3 Máme dva ukazatele do stejného pole. Jak zjistíme, který z ukazatelů ukazuje na prvek, který je v poli dříve?

- 1 Napište program, který zjistí, zda jsou dvě pole stejná (tj. mají na všech indexech stejné prvky). Můžeme předpokládat, že pole mají stejný počet prvků.
- 2 Upravte předchozí program tak, aby zjistil, zda pole obsahují stejné hodnoty (mohou být na různých indexech). Pozor, pole mohou obsahovat hodnotu i víckrát!
- 3 Zkuste odhadnout, jaký bude výstup následujícího kódu. Na kterém místě bude znak 'B'? Vaši domněnku ověřte.

```
#include <stdio.h>
```

```
int main(){  
    char retezec [] = "ABC" ;  
    char znak = retezec [2];  
    retezec [2] = retezec [1];  
    retezec [1] = retezec [0];  
    retezec [0] = retezec [2];  
    retezec [2] = retezec [1];  
    retezec [1] = znak ;  
    printf ("%s" , retezec );  
    return 0;  
}
```

- 4 Napište program, který určí, zda je zadaný řetězec palindromem (čte se stejně od začátku i od konce).
- 5 Napište program, který v zadaném řetězci spočítá počet mezer, souhlásek a samohlásek.
- 6 Zkuste odhadnout, jaký bude výstup následujícího kódu. Vaši domněnku ověřte.

```
#include <stdio.h>
```

```
int main(){  
    int pole[] = {1, 2, 3};  
    int *prvek = pole;  
    pole[0] = 2;  
    pole[1] = pole[2];  
    pole[2] = *prvek;  
    printf("%d", pole[2]);  
    return 0;  
}
```




- 7 Bez použití operátoru `sizeof()` zjistěte, jak velkou část paměti zabírají typy `char`, `int` a `double`.
- 8 Napište program, který převrátí pořadí prvků pole. Úkol vyřešte pomocí přístupu k prvkům pole přes hranaté závorky a pomocí pointerové aritmetiky. Srovnejte dobu běhu programu obou dvou přístupů.